Information Technology
and
Intelligent Transportation Systems
2003-2008
Strategic Plan Summary

Technical Appendix B
Technical Architecture Detail

January 19, 2004

Prepared for
Miami-Dade Transit

By
The Palisades Group USA

TABLE OF CONTENTS

1 TARGET TECHNOLOGY ARCHITECTURE 6

1.1 INTRODUCTION 6
1.1.1 MDT IT Technology Guiding Principles 7
1.2 TECHNICAL APPLICATION ARCHITECTURE: CLIENT / SERVER MODELS 8
1.2.1 Introduction 8
1.2.2 Monolithic Model 8
1.2.3 Two-Tier Client / Server Model 10
1.2.4 Three-Tier Client / Server Model 13
1.2.5 N-Tier Client / Server Model 14
1.2.6 Target Technical Application Client / Server Model 16
1.3 MIDDLEWARE ARCHITECTURE 17
1.3.1 Introduction 17
1.3.2 Middleware Definition 18
1.3.3 Application Integration Middleware Services 18
1.3.4 Database Middleware 18
1.3.5 Message-Oriented Middleware 18
1.3.6 Middleware Benefits 18
1.3.7 Middleware Guiding Technical Principles 19
1.3.8 Middleware Application Integration Technical Guidelines 19
1.3.9 Middleware Technical Standards 20
1.4 NETWORKING ARCHITECTURE 21
1.4.1 Introduction 21
1.4.2 Network Architecture Definition 21
1.4.3 Network Architecture Guiding Principles 21
1.4.4 Network Architecture Technical Guidelines 21
1.4.5 Network Architecture Technical Standards 22
1.5 PRODUCTION ARCHITECTURE 26
1.5.1 Introduction 26
1.5.2 Production Architecture Definition 26
1.5.3 Production Architecture: Platform Components 27
1.5.4 Production Architecture Technical Principles 28
1.5.5 Production Architecture Guidelines 30
1.5.6 Production Architecture Platform Standards 32
1.6 DEVELOPMENT ARCHITECTURE 32
1.6.1 Introduction 32
1.6.2 Development Architecture Definition 33
1.6.3 Development Architecture Principles 34
1.6.4 Development Architecture Tools 34
1.6.5 Development Architecture Platform Standards 36
1.7 OPERATIONS ARCHITECTURE 36
1.7.1 Introduction 36
1.7.2 Operations Architecture Definition 37
1.7.3 User Interface for Operations Staff 37
1.7.4 System Management Applications 38
1.7.5 Management Information Database 39
1.7.6 Operations Architecture Technical Principles 39
1.7.7 Operations Architecture Technical Guidelines 40

1.7.8 Operations Architecture Standards....................42

2 WIRELESS AND ON-BOARD COMMUNICATIONS....................43

2.1 VEHICLE AREA NETWORK....................43

2.1.1 Description....................43

2.1.2 Requirements....................43

2.1.3 Available Standards....................43

2.2 WIRELESS COMMUNICATIONS....................43

2.2.1 Description....................43

2.2.2 Requirements....................44

2.2.3 Available Standards....................44

2.2.4 Wireless Communications Issues....................44

2.3 RADIO COMMUNICATIONS....................44

2.3.1 Description....................44

2.3.2 Requirements....................45

2.3.3 Available Standards....................45

INDEX OF FIGURES

FIGURE 1-1 MONOLITHIC MODEL 9
FIGURE 1-2 A TWO-TIER FAT CLIENT MODEL 11
FIGURE 1-3 A TWO-TIER FAT SERVER MODEL 12
FIGURE 1-4 A THREE-TIER, CLIENT / SERVER APPLICATION....................14
FIGURE 1-5 N-TIER, CLIENT / SERVER APPLICATION 15
FIGURE 1-6 MDT N-TIER WITH CENTRALIZED DATA MANAGEMENT 16
FIGURE 1-7 PRODUCTION ARCHITECTURE 26
FIGURE 1-8 DEVELOPMENT ARCHITECTURE 33
FIGURE 1-9 DEVELOPMENT ARCHITECTURE SEPARATION 36
FIGURE 1-10 OPERATIONS ARCHITECTURE 37
FIGURE 1-11 OPERATIONS MANAGEMENT COMPONENTS 38
FIGURE 1-12 APPLICATION SERVER MONITORING SCENARIO....................39
FIGURE 1-13 OPERATIONS ARCHITECTURE SYSTEMS MANAGEMENT 41

INDEX OF TABLES

TABLE 1-1 MDT IT TECHNOLOGY GUIDING PRINCIPLES....7
TABLE 1-2 CONSIDERATIONS IMPACTING MDT IT TECHNOLOGY GUIDING PRINCIPLES....7
TABLE 1-3 MONOLITHIC MODEL CHALLENGES....10
TABLE 1-4 MIDDLEWARE APPLICATION INTEGRATION SERVICES....19
TABLE 1-5 TARGET NETWORK ARCHITECTURE GUIDELINES....22
TABLE 1-6 PRODUCTION ARCHITECTURE GUIDELINES....31
TABLE 1-7 PLATFORM STANDARDS....32

1 TARGET TECHNOLOGY ARCHITECTURE

1.1 Introduction

The Target Technical Architecture defines major kinds of technologies needed to provide an environment for the applications that are managing data. The Target Technology Architecture consists of the future physical infrastructure and systems software layers. The ITS Strategic Plan 2003–2008 Technology Architecture will define strategic platforms and direction, and thus provide the technical basis for further design and configuration detail.

A sound Target Architecture mitigates the complexities of development and maintenance, the obsolescence of technologies, and the possibilities that all the parts of a solution may not work together. Architecture is a proven mechanism and approach that can be used to isolate and mitigate the risks of delivering applications now and into the future. An architecture plan provides a framework for how intelligent transportation systems and information technology systems will work together to support MDT's current and future needs. It is a critical success factor in client/server and net-centric systems development.

In addition to providing a framework for making different systems work together, architecture provides differentiation to assist in "build-versus-buy" decisions. At one extreme, the IT Division can choose to "build an architecture" to make it very close to ideal; this necessitates a great deal of development that would be done by the IT Division staff. At the other extreme, the IT Division could choose to not follow technology architecture and purchase only "best of breed" applications with no assurance of integration and with expensive proliferation of incompatible technologies with significant maintenance costs. The MDT IT Division's direction has had a tendency to "buy" rather than "build" and several core application components are now in place. Within the transit industry, vendors have been slow to adopt the more sophisticated open platform methods now maturing and available in the marketplace. In most cases, transit vendor architectures remain proprietary or "closed" in nature due to extreme competitive pressures, and often do not offer any real choice of platform.

In recognition of these challenges, the MDT IT Division will move forward over the 2003–2008 planning horizon to direct project initiatives within the framework of a modern Technical Architecture, wherein open platform and sophisticated client/server models better enable Transit to serve its customer. The IT Division plans to continue its policy of building technical architecture components only when essential and not compromise in providing needed application functionality. By purchasing compatible systems rather than having to learn new hardware and software, the MDT IT Division can more easily apply its technical resources in key areas of systems engineering, data design, and data management, as well as rollout of new applications. As Transit services demand for transparent access to information increases, ITS developers can focus on sound integration of data and applications rather than the maintenance of legacy or incompatible systems.

1.1.1 MDT IT Technology Guiding Principles

The MDT IT Technology Guiding Principles relevant to the organization's business and technical strategies are listed below in Table 1-1. The IT Guiding Principles support MDT Mission / Vision and Goals. The IT Guiding Principles establish the basis for governance and implementation of the IT / ITS Strategic Plan and architectures, and affect development, maintenance, and selection of technologies.

Table 1-1 MDT IT Technology Guiding Principles

1. Data will be owned, shared, and controlled as a Transit asset.
2. Applications will be developed using a cooperative process.
3. Application initiatives will be guided by an established methodology using a systems engineering approach.
4. A technology infrastructure will be developed to facilitate integration of data and systems.
5. Secure network architectures will be employed.
6. IT will focus on and be measured by the value of solutions delivered to internal and external stakeholders and Transit customers.
7. IT will be a partner in reengineering and improving business processes.
8. IT will adhere to national and regional standards for Transit architecture.
9. IT will insure recoverability to protect the continuation of the business.
10. Apply open systems concepts to insure portability, scalability, interoperability, and compatibility of information technology systems.

IT Policies and Guidelines, the Systems Life Cycle Development, and the Capital Process are impacted by the actions driven by the IT Guiding Principles. Explicit standards-oriented policies and guidelines have been developed in compliance with these principles and are included in the remainder of this document.

Additional considerations of the MDT business environment that would impact the adoption of MDT IT Technology Guiding Principles are summarized in Table 1-2.

Table 1-2 Considerations Impacting MDT IT Technology Guiding Principles

#	Consideration	Description
1	Regional Integration	County goals for multi-modal regional transit
2	Geographical distribution	Business functions are spread over several different sites; expansion to new sites is planned.
3	Centralization of services vs. local autonomy	Ability of local sites to be able to operate independently from the central organization
4	Performance	Business improvement goals as measured by service and ridership, and by technical improvement goals as measured by response time and / or batch throughput.

5	Scalability	Plans for services expansion (PTP) and use of ITS has caused volume of data to expand, and volume of processing to increase.
6	Mobility	Business functions require workers to have access to corporate information even when they are out of the office or unable to connect to the corporate network.
7	Multiple distribution channels	New Transit services will provide customer information through a variety of communication devices.
8	Currency	Business requirements for freshness of some data will indicate that real-time or close-to-real-time data currency is necessary.

1.2 Technical Application Architecture: Client/Server Models

1.2.1 Introduction

Basic to all application models — regardless of what they do and the technology with which they are implemented — are three general areas of functionality:

- Business rules: Business rules are the parts of the business process that computer applications automate.
- Data access: Data access code automates the storing, searching, and retrieving of data by computer applications.
- Interface: The interface allows applications to communicate with applications and people.

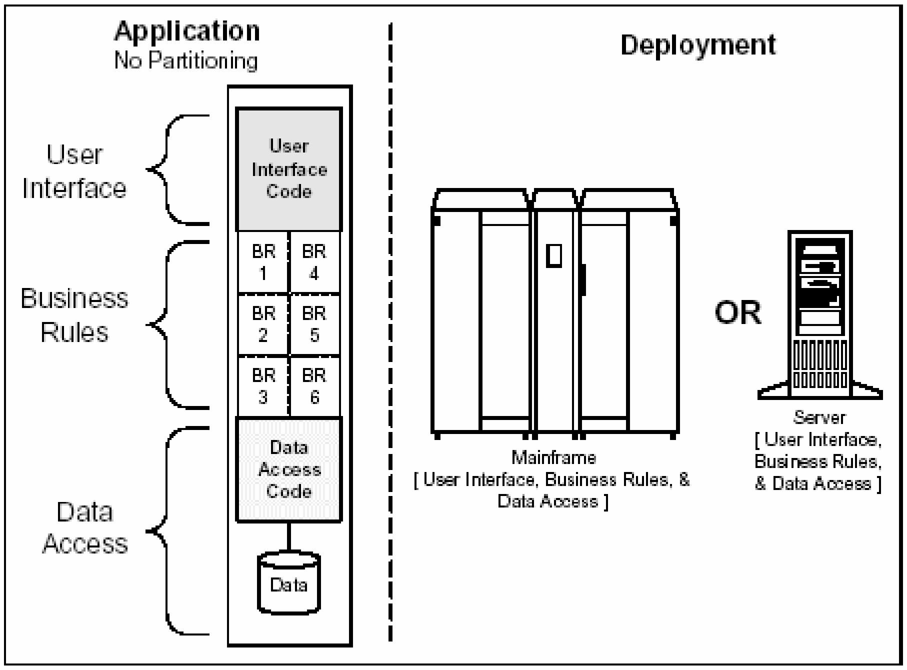
The ways in which these application functions are assembled determine:

- The flexibility of the applications
- How quickly they can be modified to support changes in business and technology
- How easily they interface with people and with each other

1.2.2 Monolithic Model

While the Monolithic Model is not a client/server deployment, it was discussed briefly, as MDT has key business processes reliant on legacy applications using this model. Monolithic applications are characterized by code that implements the business rules, data access, and user interface tightly coupled together as part of a single large computer program. Monolithic applications were typically deployed on a single platform, commonly a mainframe or midrange platform (see Figure 1-1, Monolithic Model).

Figure 1-1 Monolithic Model



(Adapted from Enterprise Wide Information Technology Architecture (EWITA) links and resources: http://www.ewita.com/)

The legacy monolithic applications are inherently limited in integration capabilities in a target networked environment that will become increasingly reliant on Internet technologies and newer generations of software tools. These challenges are coupled with support and maintenance costs associated with the underlying legacy platform which are often characterized by out-dated technology, an antiquated technical platform (e.g. DEC/VMS), multiple patches and fixes over the years, and missing or non-current documentation. Additional challenges to support MDT's monolithic applications are listed in Table 1-3.

Table 1-3 Monolithic Model Challenges

#	Challenge	Description
1	High Maintenance	Changing the code that implements a business rule has a higher risk of impacting other code in the application. When any code in the application changes, the entire application must be retested and redeployed (as opposed to only changing and testing code modules).
2	Low Integration	It is difficult to integrate applications to share services and data. Monolithic applications generally do not have well-defined interfaces that can be easily accessed by other applications.
3	Low reuse of code	Low reuse of redundant code between applications, making it more expensive to build and maintain applications. Many applications contain functionality already replicated in other applications. These applications are slower to build and test, and therefore more costly, because existing functionality is reinvented many times. These applications are also more expensive to operate and maintain over the system's life cycle, since the same data must be gathered, entered, and stored in many places.
4	Limited Communication	It is difficult to have applications communicate with other applications due to the proprietary nature of the underlying technology. Monolithic applications are not designed to communicate easily with other applications, especially over networks.
5	Limited User Interface	Monolithic applications can be accessed using only a single user interface. Many can only be accessed via terminal emulation. Having a single user interface is a limitation when application services need to be accessed from other user interfaces such as web browsers or the telephone (via VRUs).
6	Low flexibility	*There is no flexibility in where the applications can be deployed.* Applications must be deployed on a single machine capable of providing significant computing capacity to process all parts of the application: the user interface, the business rules, and the data access code.

1.2.3 Two-Tier Client/Server Model

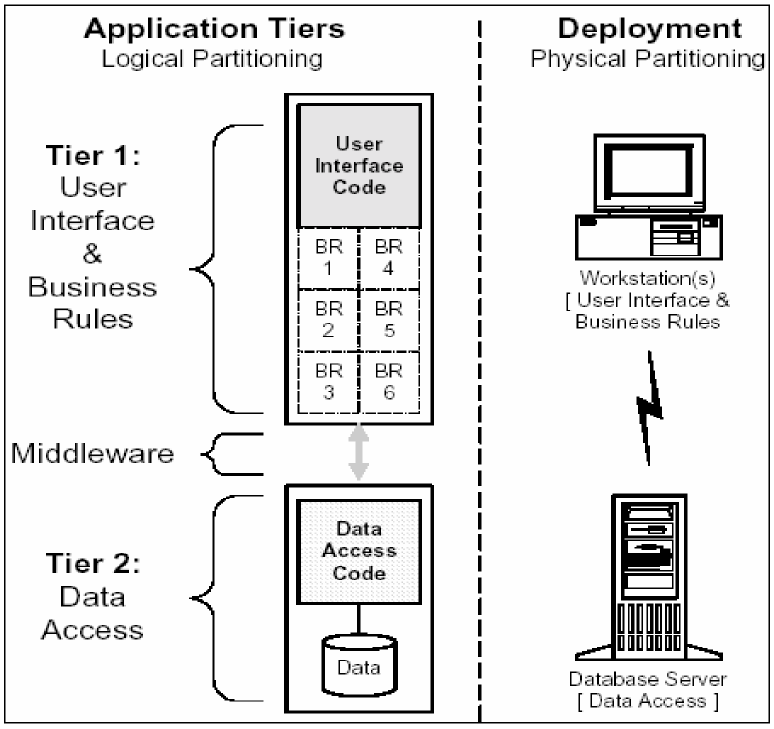
Like many organizations, the MDT IT Division has recognized the constraints of monolithic applications, and has begun the adoption of two-tier client/server technology for new applications. (At the time of this report, the MDT IT group is assessing an expanded 2-tier model using Citrix technology.) In general, client/server applications are constructed of software "clients" that, in order to perform their required function, must request assistance — "service" — from other software components, known as "servers." In many cases, proprietary protocols provide communication between client and server.

MDT has implemented of the two-tier client/server model as Two-Tier Fat Client and Two-Tier Fat Server. It was noted that in both 2-tier models, application functionality is partitioned into two executable parts, or "tiers." One tier contains both the code that implements a graphical user interface (GUI) and the code that implements the business rules. This tier executes on PCs or workstations and requests data from the second application tier, which usually executes on the machine where the application's data is stored.

1.2.3.1 Two-Tier Fat Client

This model is referred to as two-tier, fat client, because the application is partitioned into two tiers of executable code, and most of the application's code is contained in the tier executing on the workstations, known as the "fat client." (See Figure 1-2, A Two-Tier Fat Client Model.) Since business rules are tightly integrated with user interface code, the code implementing the business rules must be deployed on the same platform(s) as the user interface, and the entire workstation-resident portion of the application must be redeployed when either a business rule or the user interface changes. When the number of workstations used is high or geographically dispersed, the maintenance costs for two-tier, fat client applications escalate quickly.

Figure 1-2 A Two-Tier Fat Client Model

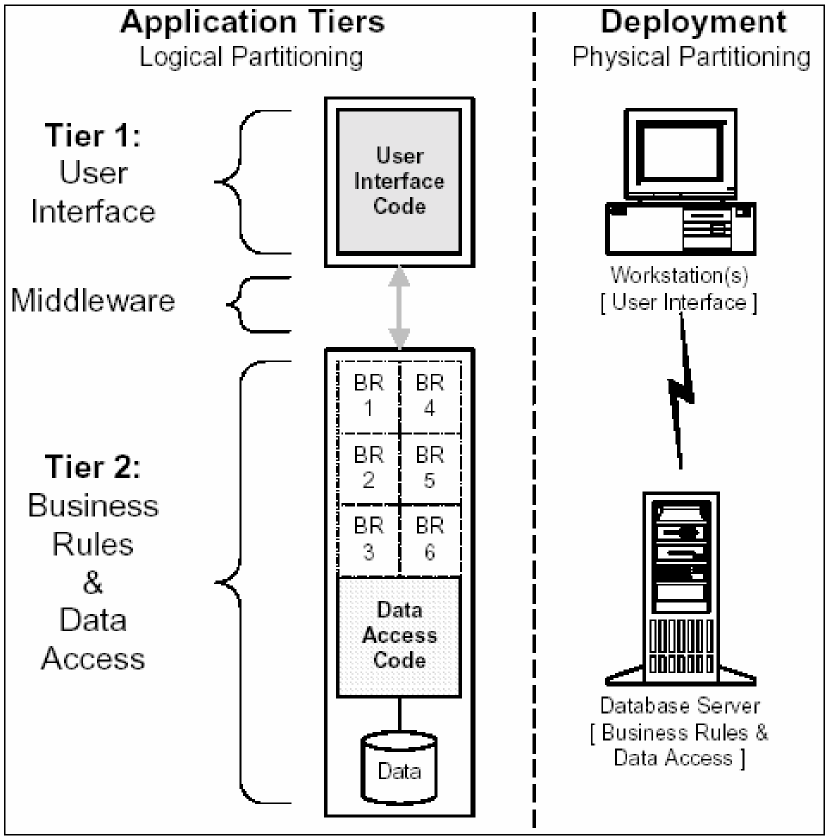


(Adapted from Enterprise Wide Information Technology Architecture (EWITA) links and resources: http://www.ewita.com/)

1.2.3.2 Two-Tier Fat Server

This model is also partitioned into two tiers, but much of the code that implements the business rules is tightly integrated with the data access code, sometimes in the form of database-stored procedures and triggers (see Figure 1-3, Two-Tier Fat Server Model.)

Figure 1-3 A Two-Tier Fat Server Model



(Adapted from Enterprise Wide Information Technology Architecture (EWITA) links and resources: http://www.ewita.com/)

The MDT IT group is moving toward two-tier, fat server applications implemented as server-centric applications that have web browsers for the user interfaces. This approach is actually a good first step in migrating to a three-tier or N-tier application architecture. Users can enjoy the ease-of-use provided by the web's graphical interface while analysts update other functions of the application.

Best practice experience with two-tier client/server applications indicate some implementation and support challenges as the business rules in two-tier application are tightly integrated with the user interface code or data access code. These challenges or drawbacks include:

- *They are difficult and expensive to modify when business requirements change.* Business rules are usually interdependent with other parts of the application program. Changing any business rule impacts the rest of the application. The complexities primarily impact response time to business change, and may also result in higher maintenance costs.
- *There is little reuse of redundant code.* It is difficult to repurpose any business rules elsewhere (e.g., in other computer applications that require similar services or in batch processing that is part of the same application).
- *There is little flexibility in selecting the platforms where the application will be deployed.* In two-tier, fat client applications, the business rules must execute on the same platform as the user interface, because the code they are implemented in is tightly coupled with the interface. Likewise, in two-tier, fat server applications, the business rules can only execute on the machine that hosts the database, because they are implemented either with the database or inside the database.
- *They can only be accessed by users with PCs running a graphical user interface.* Since the user interface is graphical, and requires a workstation to run, users with other I/O devices are excluded from using the application. These devices include existing non-graphics terminals (e.g., terminal emulation, green screens).
- *They are more difficult to manage than monolithic applications.* Any change to either business rules or GUI means that the entire workstation-resident portion of the application must be redistributed and installed on every workstation that uses the application. Frequent software distribution can be time consuming and logistically difficult to manage.

1.2.4 Three-Tier Client/Server Model

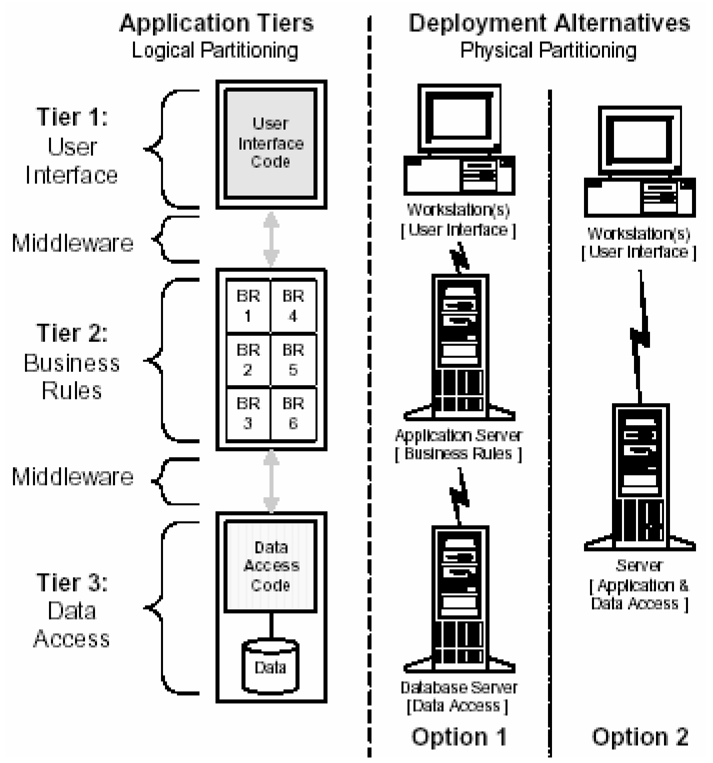
Some client/server applications are partitioned into three executable tiers of application code; user interface, business rules, and data access. It does not imply that the three tiers execute on three different platforms. Often, the business rule tier is deployed on the same platform as the data access tier; or on the same platform(s) as the user interface. There is more flexibility in where application executables can be deployed, and may be considered a good transition step from monolithic or two-tier applications.

Three-tier client/server applications still suffer from some of the limitations of two-tier and monolithic applications. Since the business rules are monolithic:

- Changes to any business rule require relinking, retesting, and redeploying the entire executable containing all business rules.
- Reduced flexibility where any given business rule can be deployed, since all business rules are tightly coupled in the monolithic tier and, therefore, must be deployed on the same platform.

This is illustrated in Figure 1-4, A Three-Tier, Client/Server Application. Notice that in the deployment, or physical partitioning, of the application the business rules are separate from both the user interface and the data access code. Business rules are deployed on their own server or on the same server as the database. Although it is also possible to deploy the business rules on the same platform as the user interface in a three-tier application architecture, it is not recommended because of the software management problems that occur with the many or dispersed user workstations that are used.

Figure 1-4 A Three-Tier, Client/Server Application



(Adapted from Enterprise Wide Information Technology Architecture (EWITA) links and resources: http://www.ewita.com/)

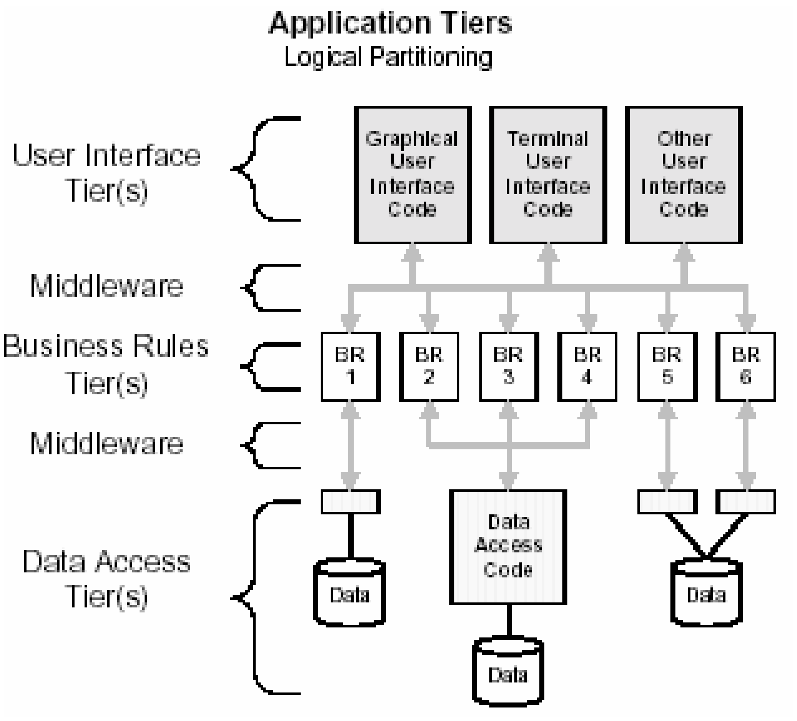
1.2.5 N-Tier Client/Server Model

Many benefits can be gained by implementing applications in the N-tier architecture. In this approach there is greater application adaptability. In the application illustrated in Figure 1-5, N-Tier, Client/Server Application, each business rule is implemented as a discrete executable (a “service”) that can be requested by any client.

Since the business rules are implemented as separate executables, any combination of business rules may run on any combination of platforms. There is flexibility in selecting the platforms where application components can be deployed. As transaction loads, response time, and throughput change, any individual service can be moved from the platform on which it executes to another, more powerful platform. Application deployment is flexible and scalable to accommodate greater transaction volumes.

Since business rules are implemented discretely, instead of tightly integrated with the graphical user interface, changes to business rules do not always require updates of code on the workstations accessing the application. It is easier to manage the deployed application. Business rules can be invoked from users accessing the application from a GUI, from character terminals, or from web browsers. They can also be accessed by telephone from VRUs or by batch jobs. A separate interface tier provides programmer productivity and consistency of application behavior.

Figure 1-5 N-Tier, Client/Server Application



(Adapted from Enterprise Wide Information Technology Architecture (EWITA) links and resources: http://www.ewita.com/)

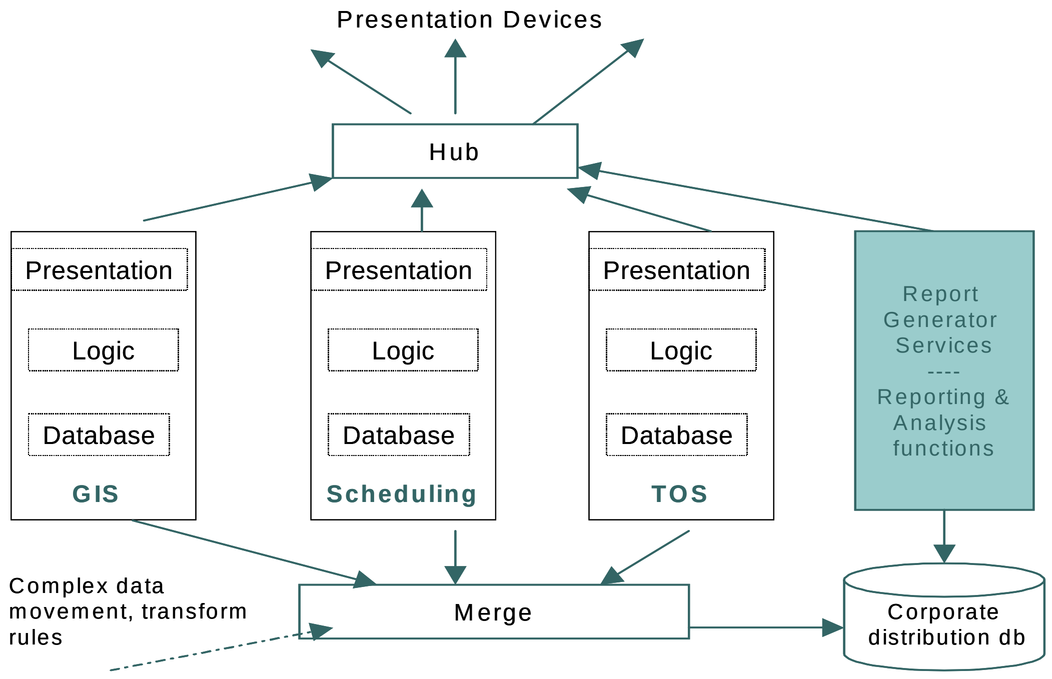
N-tier applications have the following advantages:

- It is easy to modify them to support changes in business rules.
- There is less risk modifying the code that implements any given business rule.
- N-tier applications are highly scalable.
- N-tier architecture offers the best performance of any client/server application architecture.
- They can support any combination of user interfaces: character, graphical, web browser, telephones, and others.
- They offer the highest potential for code reuse and sharing.

1.2.6 Target Technical Application Client/Server Model

The MDT Target Architecture will move in a well-planned manner toward an N-Tier Client/Server Architecture with Central Data Management. This model is illustrated in Figure 1-6, MDT N-Tier with Centralized Data Management.

Figure 1-6 MDT N-Tier with Centralized Data Management



Through this model, the MDT IT Target Architecture will provide for critical enterprise data to be managed and stored distinct from the many business processes that access and use it. Modern application development tools and technologies will be applied to address data integration and management challenges. Systems approaches will employ:

- Units of code previously duplicated in many applications that can be packaged into components or services and reused by different applications.
- Middleware integration approaches that allow applications to communicate with each other, access data residing on different platforms, and access the shared services.
- New user interface devices, such as web browsers, pagers, and voice response units (VRUs), some of which have already been introduced.
- Data standards (TCIP) based on ITS National Architecture policy.

The N-Tier Client/Server Model with Centralized Data Management is a distributed architecture environment where a key characteristic is data that is physically separated from the application processing or distributed among two or more servers, which then must be coordinated. Applying the technical principles, models, and standards expanded upon in the following sections is key to reducing the complexity and cost of the Target distributed environment. The Logical Architecture prescribes the identification and management of "core data" (see *Logical Architecture Report*). This recommended model requires the disciplined application of data warehousing concepts within a centralized data management distribution architecture framework.

An important factor to consider is that the transition process needed to deliver N-Tier distributed architecture systems solutions will likely last several years. Over the five-year planning horizon, MDT will move through evolutionary stages of distributed architecture approaches. During this period it may be expected that multiple client/server computing models will require support, and legacy systems will continue in place for some time. Migration strategies are outside the scope of this report; however, much will be dependent on available vendor solutions and resources. The move to distributed architecture solutions must be deliberate and well planned. Key to the success of the MDT IT Division is the consistent application of adopted technical principles and standards, the recognition and tradeoffs associated with client/server implementation models, and the appropriate use of integration techniques classified as "middleware".

1.3 Middleware Architecture

1.3.1 Introduction

The MDT IT Division group has the complex challenge of integrating disparate systems and islands of automation into a single enterprise-wide flow of business logic. Often, information needed by Transit business users is spread throughout various business applications in different departments within the agency. Business user workers would like to access all the information they need in a transparent and seamless fashion. To accomplish this, application developers must know how to connect information to applications and customers no matter where it resides in the network.

Middleware Architecture enables MDT IT to address these connection needs in a consistent and business-useful manner. Middleware has been described as the software glue that binds applications together across a network. Middleware can allow departments to share data between systems that do not communicate easily. Middleware is the enabler of application communications in a distributed system and is the tool that improves the overall usability of an environment made up of products from many different vendors on multiple platforms.

The middleware concept is difficult to understand from an enterprise viewpoint without having some understanding of how the introduction of the client-server environment and distributed environments affect the complexity of computer programming. Middleware vendors are trying to address some of these complexities by centralizing certain functions that may have been embedded in the tools of the network analyst, the application analyst, and the database analyst.

All agency business applications distributed over a two-tier, three-tier, or N-tier environment involve network communications between clients and servers, which require some of the functionality provided through middleware tools. Some middleware functionality is acquired within the operating systems and in database products, or obtained as separate tools and/or through functionality coded into local applications. The acquisition of an enterprise-class middleware product would enable addressing and centrally managing many middleware service needs.

1.3.2 Middleware Definition

Middleware Architecture defines the functions that enable communications in a distributed system and the tools that improve the overall usability of an architecture made up of products from many different vendors on multiple platforms. Middleware is software that allows organizations to share data between disparate systems that do not communicate easily. Middleware is often categorized into application integration middleware, database middleware, and messaging-oriented middleware.

1.3.3 Application Integration Middleware Services

Application integration middleware might be a service that enables running a legacy system through a thin-client browser or a service that enables the execution of multiple application functions from an integrated user interface. In this section, we will address the methods used to achieve this integration, including application program interfaces (API), remote procedure calls (RPC), and object request brokers (ORB). Summary guidance is provided at the end of this section.

1.3.4 Database Middleware

Database Middleware enables applications to communicate with one or more local or remote databases. It does not transfer calls or objects. For example, database middleware does not allow for two-way communication between servers and clients. Servers cannot initiate contact with clients, they can only respond when asked.

1.3.5 Message-Oriented Middleware

Message-Oriented Middleware provides an interface between applications or application parts, allowing for the transmission of data back and forth intermittently. Messaging middleware is similar to an e-mail system that transfers messages between people, except that it sends information between applications. If the target computer is not available, the middleware stores the data in a message queue until the machine becomes available. To address such complex message sending, tracking, and receipt recording requirements, MOM vendors provide several different message services.

1.3.6 Middleware Benefits

Middleware products may be used to benefit the MDT IT group in the following three ways.

- Middleware services and tools are key to creating citizen-centric service portals that enable information and services to be obtained at one place.
- Middleware services and tools are key to extending the utility of MDT's technology infrastructure and skilled workers while developing new services that rely on communications between existing services.

- Middleware services and tools are key to interfacing beyond inter-department boundaries as future requirements may include tri-county agencies and the business sector.

1.3.7 Middleware Guiding Technical Principles

The Middleware domain is guided by ITS Guiding Principles #1, #4, #6, #8, and #10:

1. Data will be owned, shared, and controlled as a Transit asset.
4. A technology infrastructure will be developed that facilitates integration of data and systems.
6. IT will focus on and be measured by the value of solutions delivered to internal and external stakeholders, and Transit customers.
8. IT will adhere to national and regional standards for Transit architecture.
10. Apply open systems concepts to insure portability, scalability, interoperability, and compatibility of information technology systems.

1.3.8 Middleware Application Integration Technical Guidelines

Middleware protocols and services related to AI Application Integration are listed in Table 1-4, Middleware Application Integration Services:

Table 1-4 Middleware Application Integration Services

Obsolescent	Transitional	Target	Emerging
Simple Object Request and Request Broker Protocols/Suites			
		MS DCOM +, distributed common object model J2EE/RMI, Java 2 Enterprise Edition (the distributed version) and Remote Method Invocation	ebXML (includes SOAP, Object Access Protocol)
Enterprise Application Integration Services (EAI)			
		Use of Integration Servers/Services	Integration Servers/ Services
Workflow Tools			
		Workflow Tools	
Remote Procedure Calls			
		DCE RPC DCE secure RPC (integrated with DCE security protocols for authentication, protection level and authorization)	XML synchronous methods SOAP
Object and Application Interfaces			
		IDL (interface definition language) stubs; MIDL/.NET (Microsoft); XML	SOAP

Please note that the terms “Obsolescent, Transitional, Target, and Emerging” are defined below to provide guidance regarding the status of specific architecture technologies. (These same terms are applied to technical guidelines in the remaining sections of this report.)

- **Obsolescent** — The MDT IT Division will not plan new deployments of this technology. The MDT IT group will develop a plan to replace this technology. This technology may be considered legacy and no longer supported.
- **Transitional** — The MDT IT Enterprise Architecture recommends other standard technologies. MDT IT may be using this technology as a transitional strategy in movement to a strategic technology. This technology may be considered legacy and no longer supported.
- **Target** — The MDT IT Division advises the use of this technology and new deployments of this technology are recommended.
- **Emerging** — MDT IT recommends only evaluative deployments of this technology. This technology may be in development or may require evaluation or further testing in a lab environment. Use of this technology may be high-risk and require higher investment due to early adoption.

1.3.8.1 Methods to Integrate Applications

Methods to integrate applications tend to be aligned with particular application development languages (e.g., C, C++, Java, etc.). The methods are often sets of standards developed by particular industry groups. However, even when two vendors deploy implementations using the same set of industry standards (e.g., DCOM—Distributed Common Object Model), their implementations may have differences and may not interoperate. This result may be due to application tool designers extending the standards or to the standards not being sufficiently specific.

The different industry protocol sets are not designed to interoperate with one another. This is not an issue as long as applications are designed to interoperate only within their intended sphere. When developers try to extend beyond the intended sphere to other applications that implement a different industry standard, either a middleware gateway must be used to provide protocol translation or the developer must employ both sets of standards in the applications that need to communicate. In general, it is advisable to use only one RPC method for within-agency distributed functions. More detail on integration methods with middleware is included in Appendix E: *Middleware Technical Detail.*

1.3.9 Middleware Technical Standards

At the time of this report, MDT is expecting to implement back-end business systems, which require close coordination with county administrative functions, following county technical standards and direction (DCOM/. NET). Transit-specific operations systems are often dictated by vendor-provided solutions, and require support of dual platforms utilizing either DCOM/.NET or J2EE/RMI. The MDT IT Division has adopted both DCOM/. NET and J2EE/RMI as viable platform strategies. The development language selection influences object model selection so a blended middleware technology standard will be supported and specific product standards will be prescribed through a committee recommendation.

1.4 Networking Architecture

1.4.1 Introduction

The MDT IT Division promotes a single reliable, scalable, and resilient set of agency network infrastructures that economically supports Transit business functions in an efficient and effective manner. The Network Architecture provides the framework and foundation to enable MTD business processes, new transit services opportunities, and new methods for delivering these services.

The Target Network Architecture addresses all relevant criteria on a broad scale, rather than as part of the deployment of an individual application that can limit or eliminate certain options for future network components or services. The development of the Target Network Architecture is a continuous process, which is critically important in an environment where funding to implement it may not be immediately available. The ongoing process provides the opportunity to continually refine the Target Network Architecture to keep it aligned with business strategies and requirements, emerging standards, and changing technology.

1.4.2 Network Architecture Definition

The Network Architecture defines common, industry-wide, open-standards-based, interoperable network infrastructures providing reliable and ubiquitous communication for MDT's distributed information processing environment. It defines various technologies required to enable connections among its business units and counties as well as the private business sector.

1.4.3 Network Architecture Guiding Principles

The following ITS Guiding Principles, #4, #5, #6, #8, #9, and #10, guide this domain.

4. A technology infrastructure will be developed that facilitates integration of data and systems.
5. Secure network architectures will be employed.
6. IT will focus on and be measured by the value of solutions delivered to internal and external stakeholders, and Transit customers.
8. IT will adhere to national and regional standards for Transit architecture.
9. IT will ensure recoverability to protect the continuation of the business.
10. Apply open systems concepts to insure portability, scalability, interoperability, and compatibility of information technology systems.

1.4.4 Network Architecture Technical Guidelines

The technical components of the Target Network Architecture are presented relative to the OSI 7498-1 Network Reference Model in Table 1-5: Network Architecture Technical Guidelines.

Table 1-5 Target Network Architecture Guidelines

Target Network Architecture Table		
Obsolete, Transitional	**Target**	**Emerging**
OSI Layer 1 - Physical		
Coaxial cabling, Category 3 unshielded twisted pair (UTP), shielded twisted pair (STP), and 62.5 / 125-micron multimode fiber.	Category 5e UTP (supersedes Category 5 UTP), 50 / 125-micron multimode fiber, 6 / 125-micron single mode fiber. Logical star topology, SONET, ISDN / PRI, xDSL, cable modem protocols	Category 6 UTP, wireless. Logical meshed star topology.
OSI Layer 2 – Data Link		
Single-segment LANs, separate dedicated networks for different services (e.g., voice and data), separate dedicated networks for various user groups, proprietary protocols (DECnet), FDDI, X.25, time-domain (channelized) protocols (e.g., SDLC, HDLC).	Open-standards-based, multiservice networks; 100 / 1000 Ethernet; 802.11 LAN, 802.16 MAN Wireless Ethernet; Frame Relay; ATM.	Packet- and cell-based wireless, dynamic data-link level switching, and prioritization, 10G / 40G Ethernet.
OSI Layer 3 – Network		
Separate dedicated networks for different services (e.g., voice and data), flat designs with unmanaged bridges, hubs, proprietary protocols (e.g., IPX, DECnet).	Converged networks with prioritization for all services; switched, multisegment design; IP; RIP; BGP; OSPF; IP switching; and DHCP.	Dynamic, network-level switching and prioritization. VoIP, H.323
OSI Layer 4 – Transport		
Fixed IP addressing and proprietary protocols e.g., SPX	Converged networks with prioritization for all services, TCP, and UDP.	IPv6 and dynamic transport-level switching and prioritization.
OSI Layer 5 – Session		
DEC-dns.	DNS	Dynamic, session-level switching and prioritization.
OSI Layer 6 & 7 –Presentation and Application		
DEC-lat.	SNMP; RMON; SMTP.	Dynamic, content-level switching and prioritization.

1.4.5 Network Architecture Technical Standards

The following Network Architecture Standards are prescribed to coordinate the MDT IT Division’s implementation of network infrastructure. The goal is to employ only open systems based on common, proven, and pervasive industry-wide, approved, open standards; however, a full complement of open standards does not yet exist for all components of network infrastructure. Therefore, combinations of open standards, de facto industry standards, and mutually agreed upon product standards are currently required to support MDT's heterogeneous operating environment.

Network Standard NS1: The standard for copper network cabling is Category 5e Unshielded Twisted Pair (UTP).
Rationale:

- Category 5e UTP is certified to carry 100/1000 Mbps of data.
- Category 5e UTP supersedes Category 5 UTP for new installations.
- Category 5e UTP is an industry-standard structured cabling system and has support of the Institute of Electrical and Electronics Engineers (IEEE).
- UTP shall be used unless specific issues exist, such as high Electromagnetic Interference (EMI) or long transport distances.
- Wiring, cable, connector, and equipment vendors have standardized on Category 5e UTP.
- Installation of copper network cabling shall conform to applicable building codes, IEEE, EIA/TIA, and BICSI.
- Category 6 UTP specifications are currently approved by TIA and ISO working groups.

Network Standard NS2: The standards for fiber network cabling are single-mode and multi-mode, depending on requirements.
Rationale:

- Intra-building fiber network cabling may be either multimode or single-mode. 62.5/125-micron multimode fiber is capable of transmitting Gigabit Ethernet up to a distance of approximately 550 meters. 50/125-micron multimode fiber is capable of transmitting Gigabit Ethernet up to a distance of approximately 600 meters. 6/125-micron single-mode fiber is capable of transmitting Gigabit Ethernet up to a distance of approximately 100 kilometers.
- Inter-building fiber network cabling is multimode to allow Gigabit Ethernet and above transmission rates over greater distances.
- All fiber network cabling shall be open, industry-standard as supported by IEEE.
- Installation of fiber network cabling shall conform to applicable building codes, IEEE, EIA/TIA, and BICSI.

Network Standard NS3: The standards for wireless network connectivity are IEEE 802.11a/b/g (LAN) and IEEE 802.16 (MAN) with the 802.1x security standard.
Rationale:

- IEEE 802.11g offers relatively high-speed (11Mbps and 54 Mbps) links.
- IEEE 802.16 offers Metropolitan Area Networks with up to 66 GHz of performance.
- The majority of wireless manufacturers have adopted the IEEE 802.11i (check) standard combined with VPN and RADIUS services for improved security manageability.

Network Standard NS4: The logical network topology shall be a star. The physical network topology may be a star, ring, or mesh.
Rationale:
- Star, ring, and mesh topologies provide the capability to easily add or remove network devices as necessary.
- Star, ring, and mesh topologies minimize the effect of connection failures between devices.

Network Standard NS5: The standard for network link layer access protocol is Ethernet, IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD).
Rationale:
- Ethernet is a widely accepted, reliable, stable protocol supported by manufacturers.
- Ethernet is scalable; faster versions are emerging to manage the increase of data flow.
- 100/1000 Ethernet has the bandwidth necessary to support the requirements of converged voice, data, and video applications.

Network Standard NS6: The standards for transport and network protocols are TCP/UDP and IP, respectively.
Rationale:
- TCP/UDP and IP make up an open protocol suite that allows Internet access and the seamless integration of Intranets, Extranets, VPNs, and LANs.
- The majority of manufacturers support TCP/IP and TCP/IP-enabled products.

Network Standard NS7: Network devices (routers, switches, firewalls, access servers, etc.) shall be manageable with Network Management platforms that use Simple Network Management Protocol (SNMP) and Remote Network Monitoring (RMON).
Rationale:
- SNMP is part of the TCP/IP protocol suite.
- SNMP and RMON facilitate the exchange of management information between network devices.
- SNMP allows for network performance management, isolation and analysis of network problems, and growth planning.
- The majority of vendors support SNMP and RMON.

Network Standard NS8: Switching (Layers 2, 3, and 4) technologies are the standard for Local Area Network (LAN) network device connectivity.
Rationale:
- Switching at OSI Layers 2, 3, and 4 in a LAN improves network performance by enabling the balancing of network traffic across multiple segments, thus reducing resource contention, providing scalability, and increasing throughput capacity.
- Switching at OSI Layers 2, 3, and 4 in a LAN provides enhanced security and network management.

Network Standard NS9: Internal workstation network IP addressing shall be assigned using Dynamic Host Configuration Protocol (DHCP). Core infrastructure (server, network, printers, etc.) and IP management and addressing will be statically assigned and manually managed.

Rationale:

- DHCP provides flexibility for growth and migration of networks.
- DHCP facilitates and simplifies IP network administration and the addition of workstations and devices to networks.
- DHCP address allocation may be (1) an automatic allocation where DHCP assigns a permanent IP address to the workstation; (2) manually allocated and assigned by the DHCP administrator; or (3) dynamically allocated where DHCP assigns an IP address to a workstation for a limited period of time (lease).
- The majority of vendors support DHCP.
- IP addresses can be reserved from the DHCP scope for core servers, network devices and printers.

Network Standard NS10: Public announcement network services (e.g., station signage) will be implemented as a physically separate and outsourced managed service.
Rationale:

- Signage devices are planned to include advertisements and commercial offerings.
- Secure access must be assured to allow business partners ability to dynamically modify advertisements and messages.

Network Standard NS11: VPN network services will be implemented and leveraged to integrate, where needed, secure business sector access to MDT enterprise services.
Rational:

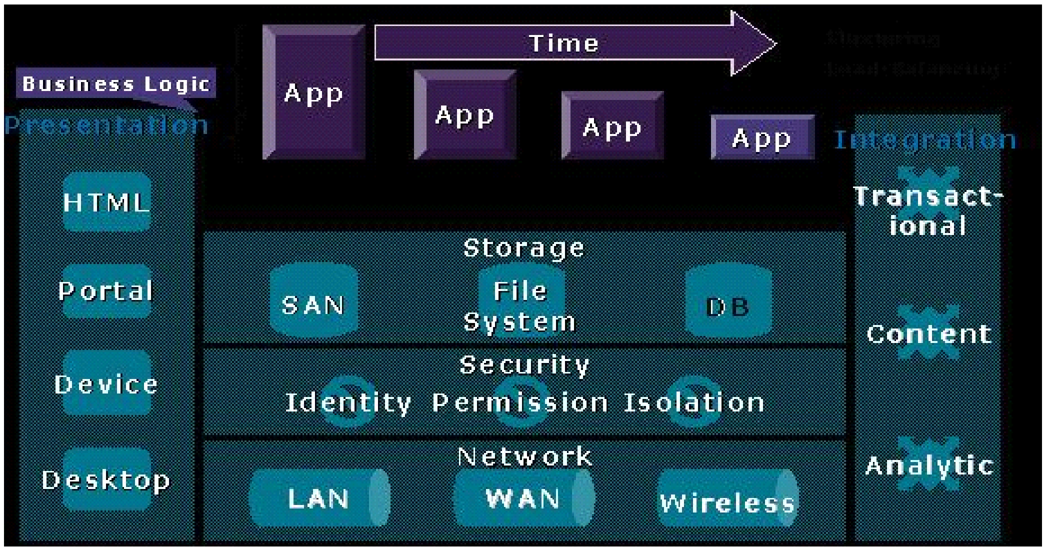
- VPN network access provides secure and ubiquitous network access services allowing flexibility and timely deployment.
- Strong security capabilities including data encryption and point-to-point control ensure MDT's enterprise network security will not be compromised while supporting the need of the business.

1.5 Production Architecture

1.5.1 Introduction

MDT IT Production Architecture describes a reliable, scalable, interoperable set of technical and logical technology platform devices to implement and support agency business functions in an efficient and effective manner. The MDT IT Production Architecture encompasses all the domain architectures contained in this document.

Figure 1-7 Production Architecture



(Adapted from The Meta Group, “The Challenge of Adaptive EAI Infrastructures.” File 762, 28 July 1999.)

Target Production Architecture is vendor/manufacturer neutral and strategic by design. As such, it does not attempt to identify or address specific device or operating system manufacturers or distributors. An individual selection of specific manufacturers, hardware configurations, and operating systems is based on the business needs of a given project, a positive business case, or the overall business requirements of the agency.

The following section supports the Production Architecture for fixed-end systems, and focuses on the establishment of standard systems platforms. On-Board Architecture is discussed in later sections.

1.5.2 Production Architecture Definition

The Target Production Architecture addresses platform devices relative to their versatility, capability to seamlessly interoperate with other platform devices, operating systems, embedded security, adherence to open or pervasive industry standards, provision for open system standard interfaces, and utilization of open-standard drivers. This approach focuses on the functionality of platform technologies to support agency business requirements rather than addressing attributes such as specific platform or vendor configurations. Further discussion on platforms follows.

1.5.3 Production Architecture: Platform Components

Production Architecture platform categories range from enterprise-class midrange servers to individual workstations and hand-held computing devices along with the operating systems that control these devices. Production platform categories include the following:

- Server — The server, with its associated operating system, provides services requested by clients. Types of servers included are: mainframes, midrange, and network servers (application, file, print, database, etc.). Servers should be positioned to embrace a variety of applications so that, over time, as open-standard operating systems and open-standard interfaces are deployed, the traditional boundary lines between voice, data, and video are eliminated. Server-attached or network-attached output devices such as printers, plotters, etc. should use IEEE-standard interfaces and industry de facto standard software drivers.

- Storage — Storage is increasingly recognized as a distinct resource, one that is best managed separately from the devices (servers, clients) that are its consumers and beneficiaries. Such storage is increasingly shared by multiple servers/clients and is acquired and managed independently from them. Storage solutions should address MDT's requirements for short term, long term, and permanent storage of information. Types of storage include:

    - Direct Attached Storage (DAS) is comprised of interfaces (controllers) and storage devices that attach directly to a server or a client. DAS, in the form of independent storage devices, RAID arrays, or tape libraries, is the most common storage architecture today.
    - Network Attached Storage (NAS) is an open-industry-standard, file-oriented storage implementation where storage devices are connected to a network and provide file access services to server and client devices. An NAS storage element consists of an engine, which implements the file services, and one or more devices on which data is stored. By connecting directly into a network, NAS technologies allow users to access and share data without impacting application servers.
    - Storage Area Network (SAN) is an open-industry-standard, data-centric storage implementation that traditionally uses a special-purpose network that incorporates high-performance communication and interface technologies as a means to connect storage devices with servers.

- Client — The client, with its associated operating system, provides the end-user interface to the business application. Clients include the personal computer (PC), thin client, host-controlled devices (terminals, telephones, etc.), voice interface devices, single- and multi-function mobile devices (Pocket PC, PDA, PDA-phone, etc.), telephony devices, smart cards, etc. "Personal" input devices (tablet, keyboard, probe, etc.) and output devices (monitors, displays, projectors, speakers, printers, etc.) attached to a client device should use IEEE-standard interfaces and industry de facto standard software drivers.

1.5.4 Production Architecture Technical Principles

The Target Production Architecture relies on sound technical principles to guide the evaluation, planning, design, selection, and implementation of platform technology and services. In this all-encompassing domain all ten MDT IT Technical Guiding Principles are applied.

1. Data will be owned, shared, and controlled as a Transit asset.
2. Applications will be developed using a cooperative process.
3. Application initiatives will be guided by an established methodology using a systems engineering approach.
4. A technology infrastructure will be developed that facilitates integration of data and systems
5. Secure network architectures will be employed.
6. IT will focus on and be measured by the value of solutions delivered to internal and external stakeholders, and Transit customers.
7. IT will be a partner in reengineering and improving business processes.
8. IT will adhere to national and regional standards for Transit architecture.
9. IT will insure recoverability to protect the continuation of the business.
10. Apply open systems concepts to insure portability, scalability, interoperability, and compatibility of information technology systems.

Because of the many complexities of the Production Architecture, additional principles specific to this domain are included below. These are labeled PP.1, PP.2, etc. to assist in differentiating them from the IT General Guiding Principles or other domain-specific principles described in this document.

Production Principle PP.1: Production Architecture platforms provide the device infrastructure to support MDT's business and administrative processes.
Rationale:

- Platforms (servers, storage, and clients) must accommodate new and expanding applications; increased storage, access, and retention requirements for a variety of data (e.g., voice, data, image, and video); and a variety of concurrent users, regardless of location.
- Specific platform choices or preferences should not dictate or restrict agency business and administrative application solutions. The platform decision is dependent on the overall application solution in accordance with agency business requirements.
- Stored information must be readily accessible from any point in the network, presented in a consistent framework and timely manner so that business decisions can be based on up-to-date information.

Production Principle PP.2: Servers and storage that support essential business processes and mission-critical business operations must be operational, reliable, and available 24x7x365.
Rationale:

- Sufficient reliability, redundancy, and fault tolerance must be built in to ensure that a single point of failure does not have severe adverse effects on essential business processes and mission-critical business operations.

- Server and storage platforms that support essential business processes and mission-critical business operations should be designed to permit continued operations during a failure that occurs in the course of normal operations, or in the event of a disaster. Throughput may be impacted during the event; however, information and services should remain accessible and available to citizens and end-users, regardless of location in the network.
  - Storage solutions should address MDT's requirements for short term, long term, and archive storage of information.
  - Timely, well-documented, and tested backups of software and information will allow MDT to recover quickly and effectively from potential interruptions of service, and also provide the requirement to restore critical information as needed.

Production Principle PP.3: Production operating system security should be based on industry-wide, open standards.
Rationale:
- Production operating system security that utilizes open standards facilitates application portability and integration across platforms.
- Security services already exist for many common operating systems and applications; however, products from different vendors may be implemented in ways that make it difficult to integrate these products into a managed security environment. Existing application, operating system, or platform security mechanisms should be used whenever they match Security Policy target standards. Application-specific security mechanisms should only be developed where necessary.

Production Principle PP.4: Production configurations and associated operating system versions should be minimized.
Rationale:
- Reducing uniqueness in platform and operating system selection and increasing standardization reduces support and maintenance costs.
- Standardization of platforms and associated operating systems can lead to increasing purchasing economies of scale for MDT.
- Standardization of platforms and associated operating systems reduces future upgrade and migration issues.
- Standardization of platforms and associated operating systems simplifies training, learning curves, and skills transfer.

Production Principle PP.5: Production infrastructure should be designed for growth, flexibility, and adaptability.
Rationale:
- Changing business processes and requirements drive application, network, and Production architecture.
- Scalable, flexible, and adaptive platform and network infrastructure facilitates the delivery of applications resulting from changing business requirements.
- As new processes are developed and new information becomes available, platform infrastructure and networks must scale to allow for increased demand.

Production Principle PP.6: Production infrastructure should maximize the design and availability of Target Network Architecture for delivery of applications and services to citizens and end-users, regardless of location.

Rationale:

- Production Architecture provides the end-user or citizen an interface to MDT information and resources. It enables the rapid, effective execution and processing of a wide spectrum of business applications. Production Architecture works in conjunction with Network and Security Policy to extend resources and information capabilities and access, regardless of the location of the citizen or end user.
- Target Network Architecture defines industry-wide, open-standards-based, scalable, flexible, and adaptive networks to facilitate the delivery of applications and services.
- Networks enable access to a wide spectrum of information, applications, and resources, regardless of the method of delivery or the location of the citizen or end user.

1.5.5 Production Architecture Guidelines

In addition to the Production Architecture Domain Principles listed above, domain-specific Guidelines and Standards will assist in the coordinated implementation of a standard platform infrastructure. The goal is to utilize platforms based on common, proven, and pervasive industry-wide, approved, open standards; however, a full complement of open standards does not exist for all components of platform architecture. Therefore, combinations of open standards, industry de facto standards, mutually agreed upon product standards, and best practices are currently required to support the MDT's heterogeneous operating environment.

A table summary of the technical components of the Production Architecture Guidelines is presented relative to the OSI 7498-1 Network Reference Model in Table 1-6, Production Architecture Guidelines.

Table 1-6 Production Architecture Guidelines

Target Production Architecture Guidelines		
Obsolete, Transitional	**Target**	**Emerging**
OSI Layers 1 – Physical, 2 – Data, 3 – Network, 4 – Transport		
Platforms that employ proprietary gateways, as opposed to open-standard interfaces	SCSI, iSCSI TCP/IP protocol x86, RISC Single application smart cards	Trusted Platform Multifunction smart cards
Platforms having proprietary operating systems without open-standard interfaces and drivers. For example: • Digital or analog PBXs requiring a separate network infrastructure • Voice mail systems without open APIs Platform and Network Operating Systems that are unable to utilize a converged network infrastructure to access business applications	Platforms having open-industry standard operating systems, with imbedded security, and open-standard interfaces and drivers Platforms having industry de facto standard operating systems, with imbedded security, and open-standard interfaces and drivers. For example: • Midrange systems TCP/IP, SIP, Open APIs • Servers with TCP/IP, SIP, Open APIs • IP telephony systems with TCP/IP, SIP, Open APIs • Storage Area Networking with multi-use access channels • Client devices (PCs, Network Computers, PDAs, etc.) with wired/wireless connectivity, TCP/IP and multifunction applications • Platforms having niche proprietary operating systems, with imbedded security, and open-standard interfaces and drivers (requires exceptional business requirements) LDAP Directory Services SNMP Management of platforms Platforms deployed on target networks, with class of service (CoS) and quality of service (QoS) availability CORBA SIP, SDP, SAP, RTSP HTML, XHTML, XML Open API Java VM MS .Net Open database/file connectivity: SQL, ODBC, PSQL, NFS, JDBC	Platforms having open-industry standard operating systems, with imbedded security, multifactor authentication, and open-standard interfaces and drivers

1.5.6 Production Architecture Platform Standards

The following table describes the platform standards identified as key systems which operational excellence and capabilities will be developed to support.

Table 1-7 Platform Standards

Tier	Storage	Scalability	Availability	OS	Middleware
Web Server	Internal Disk Storage	Scale-out Farm 1-2 CPU's 1U Servers and Blades Network-based load balancing	Redundant Stateless farm Network-based load balancing Stateless connections	Win2000 .Net Tru64 Unix	IIS Apache
App Server	SAN NAS	Scale-out farm 2-4 CPUs Rack, Blades, or standalone	Farm with failover Persistent state pushed into DB Load balancing in Middleware	Win2000 .Net Tru64 Unix	MS .Net Oracle AS
DBMS Server	SAN	Scale up Single instance Large SMP (4-16 CPUs)	OS Clustering, 2+ nodes Redundant storage DBMS clustering	Win2000 .Net Tru64 Unix	Oracle SQL Server

1.6 Development Architecture

1.6.1 Introduction

The Development Architecture must support all tasks involved in development (analysis, design, construction, testing and maintenance), and requires the same support as a similarly sized end-user execution environment.

Successful implementation of an IT project requires certain basics to be defined and in place, such as a development environment, a Development Architecture, and a systems engineering methodology that encompasses a disciplined development and testing process, organization, and tools. The Development Architecture thus focuses on the support of productivity improvements for ITS staff and the development of reliable projects that meet specifications.

Organization refers to the way a project is organized as well as staff roles, skills, and training. Actual skills and staff roles determine many decisions made about the development environment. A project development methodology helps define needed organizational components of a project and the desired skills of the project participants. Many of these organizational issues will be addressed in Task 6 of the IT/ITS Strategic planning process.

Development tools, coupled with disciplined planning and maintenance, target efficient and effective coordination of development efforts including quality assurance. Tools may automate parts of the development process and aid in the creation of code deliverables, act to leverage activities, and:

- Provide analysis to reduce uncoordinated change to same module
- Eliminate reentry of source code due to accidental deletion
- Assist repeated design, coding, testing, and maintenance of very similar logic
- Support migration to system test because impact analysis for a change request was incomplete

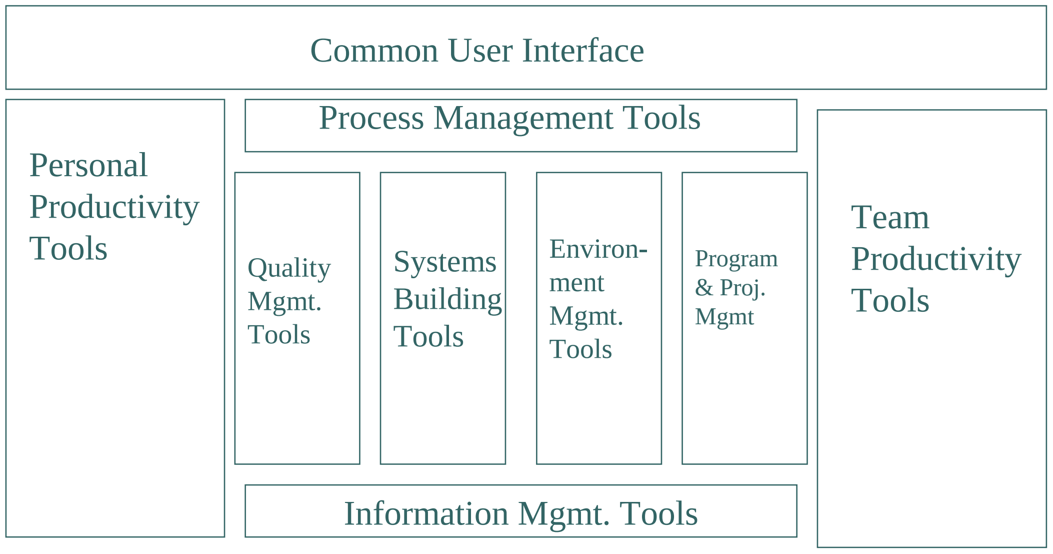
The cost/complexity of the Development Architecture is dependent on the platforms chosen for production support of business systems, the complexity of the target production environment, the size of the system being developed and maintained, project or multiple project schedules, and size/location of team resources assigned for coding, test, and implementation.

1.6.2 Development Architecture Definition

The Development Architecture is a combination of development tools, methods, standards, and procedures that define a development environment. This is depicted in Figure 1-8, Development Architecture.

Figure 1-8 Development Architecture

(Adapted from Goodyear, *Enterprise System Architectures: Building Client/Server and Web-Based Systems*, 2000.)



1.6.3 Development Architecture Principles

The Development Architecture relies on sound technical principles to guide the design, development, and quality assurance testing from unit through systems testing, migration planning, and post-rollout support. As in the Production Architecture, all ten MDT IT Guiding Principles apply.

1. Data will be owned, shared, and controlled as a Transit asset.
2. Applications will be developed using a cooperative process.
3. Application initiatives will be guided by an established methodology using a systems engineering approach.
4. A technology infrastructure will be developed that facilitates integration of data and systems.
5. Secure network architectures will be employed.
6. IT will focus on and be measured by the value of solutions delivered to internal and external stakeholders, and Transit customers.
7. IT will be a partner in reengineering and improving business processes.
8. IT will adhere to national and regional standards for Transit architecture.
9. IT will insure recoverability to protect the continuation of the business.
10. Apply open systems concepts to insure portability, scalability, interoperability, and compatibility of information technology systems.

1.6.4 Development Architecture Tools

At a high level the Development Architecture represents the various types of tools required in systems development and management along with common capabilities to integrate these tools. These include:

- **Common User Interface:** For all tools.

- **Process Management Tools:** Provide structure and control over the development process. Products should have customization capability to tailor the tool to support a specific methodology or organization of a project, platform support for current and planned platforms, and some workflow sophistication.

- **Repository:** The communication backbone for the development environment, to facilitate sharing of different processes such as design, construction, and maintenance information. Consistency is important and may be provided in common templates for content and format. Because client/server solutions occur in many different "pieces," it is essential to understand the dependencies among the pieces and to provide impact analysis to change. Repositories also allow traceability and reuse.

- **Personal Productivity Tools:** Used for a variety of single-user activities needed in project development. These include spreadsheet, graphics, and word processor.

- **Quality Management Tools:** Used to ensure an agreed level of quality in the system under development. Measurement is considered important to quality management as it provides a method for testing whether a product meets any given criteria. Quality management tools facilitate the capture of such metrics and exist to help fine-tune the development process.

  Some examples are:
  - Average number of defects at start of application constructions
  - Average number of defects at first migration to production
  - System availability and causes of downtime
  - Time needed for a user to learn the system
  - Maintainability in terms of time to fix a function or add a new function

- **Systems Development Tools:** Used by the majority of the development team(s) to capture requirements and functional design through detailed coding and testing. There are many tools available in this category and they include analysis and design tools such as data, process and event modeling, reverse engineering tools, testing tools, and configuration management tools. Many CASE products integrate meta modeling, database design, database construction, and code generation. The Logical Architecture introduces UML (Unified Modeling Language) based tools, which are gaining wider acceptance in the marketplace for distributed computing systems development. Database platform decisions drive tool selection; some though not all tools can support the creation of schemas for multiple database products.

- **Environment Management Tools:** Assist the management of the development environment as a production environment including monitoring performance, providing Help Desk support, managing changes to the distribution environment, and managing development needs for capacity. These include configuration management tools, which provide version control and migration control.

- **Program and project management tools**: Aid, schedule, and track progress against the project plan. MDT is currently using Microsoft Project for these capabilities.

- **Project Communication Tools:** Facilitate communications within and across development teams, and include e-mail, groupware, and publishing tools.

The above is a useful checklist as rarely can all capabilities be employed in initial projects; rather developers will iterate toward more complete implementations of the development model. Tight time frames and large teams will require more complete implementation where the integrating of project methodology, organization, tools, standards, and procedures is critical. If one of these falls out of synchronization with the others, lost time, inconsistent results, and project management headaches ensue.

Figure 1-9 Development Architecture Separation

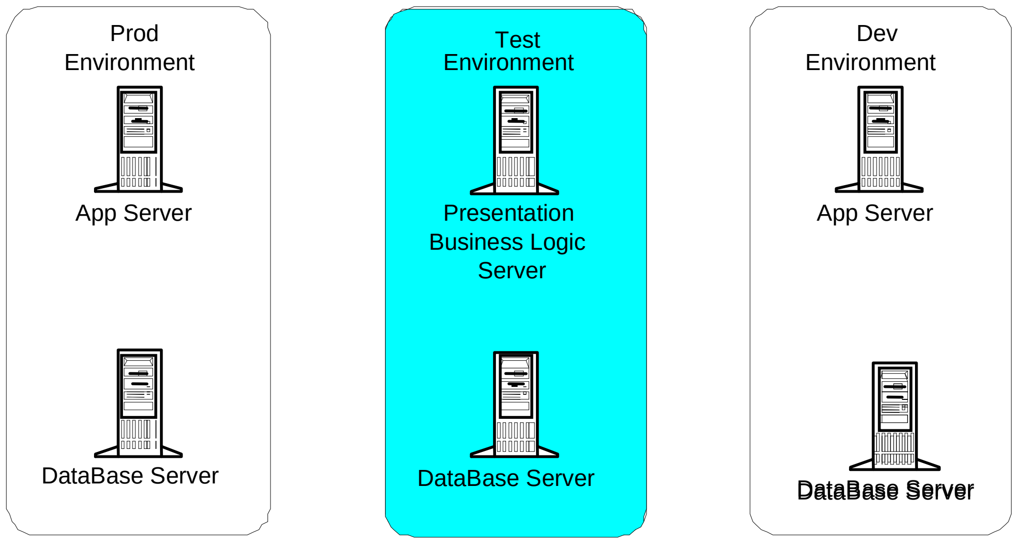


Figure 1-9 illustrates physically where common applications can be pooled onto a shared infrastructure. Clearly defined development and test environments are physically maintained separately with individualized security and operational characteristics.

1.6.5 Development Architecture Platform Standards

The development architecture environment is a mirror of the platforms used in the Production Architecture. Within the development environment, MDT is considering the adoption of tools that will further aid staff productivity. Also see Section 9.7: Production Architecture: Platform Standards.

1.7 Operations Architecture

1.7.1 Introduction

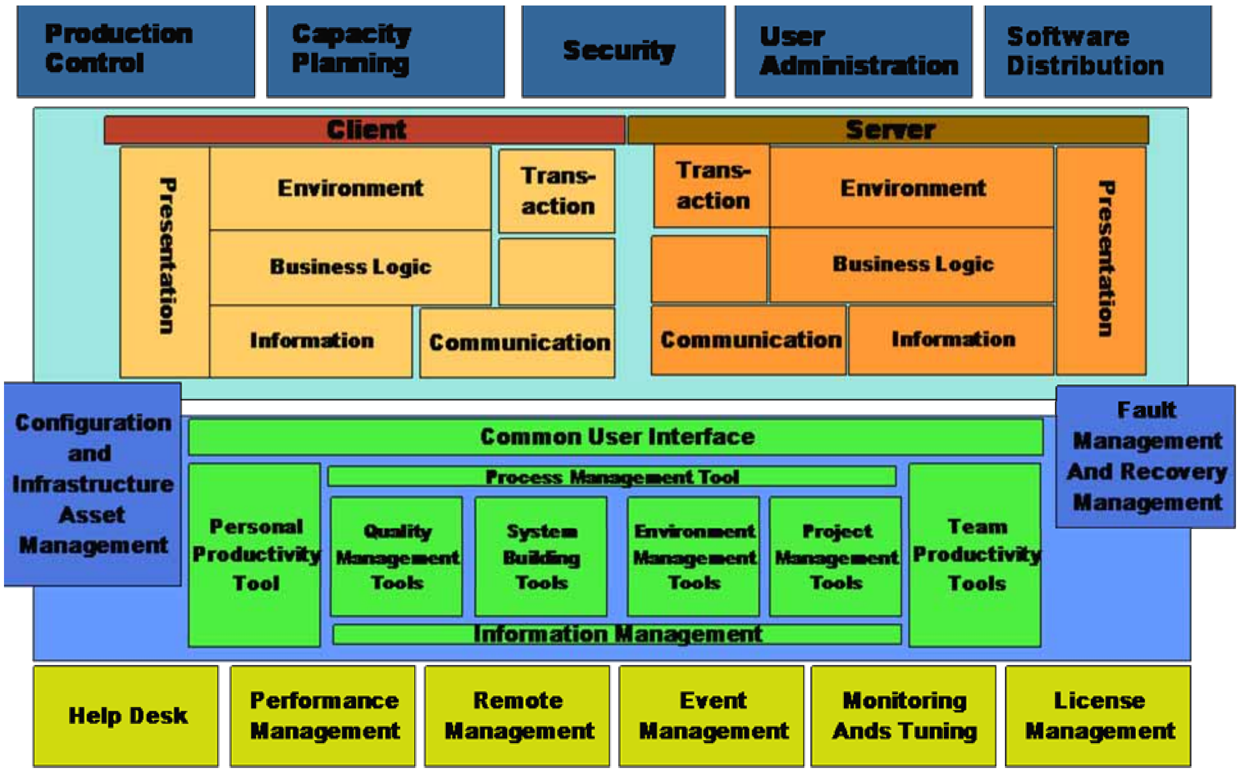
The Operations Architecture encompasses the coordination of system and network resources throughout the enterprise. The Operations Architecture provides the guidelines for managing the reliable enterprise-wide operation of mission-critical applications.

Systems management of operations in a distributed computing environment is much more complex than in host-based computing environments. Client/server systems are composed of computer nodes, networks, and applications. These three elements are logically integrated but they can be physically dispersed. The level of complexity escalates when the various components are heterogeneous. The reliability of a distributed system is dependent on the reliability of each component. Therefore reliable availability for mission-critical applications is the primary operational objective. Reliable system availability can mean 24 hours per day/ 7 days per week for some applications.

1.7.2 Operations Architecture Definition

The following Operations Architecture illustrates the many areas of the Operations Architecture.

Figure 1-10 Operations Architecture



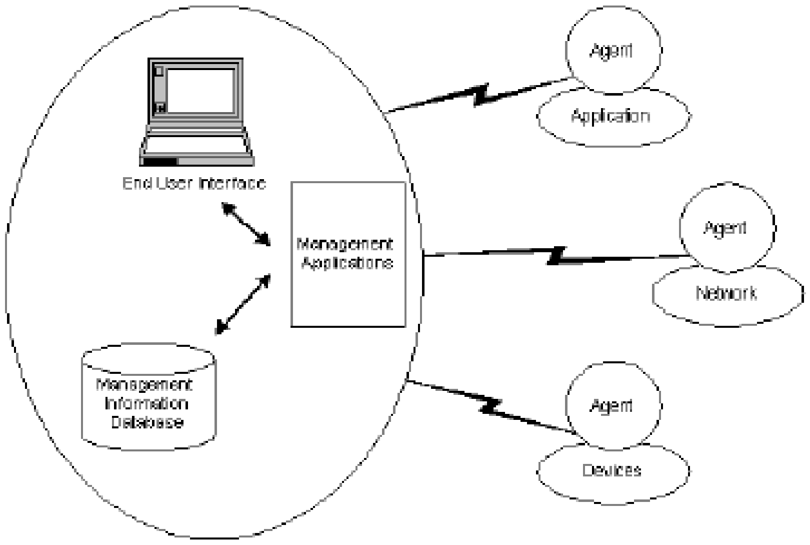
(Adapted from Goodyear, *Enterprise System Architectures: Building Client/Server and Web-Based Systems*, 2000.)

The following components are identified as necessary for the successful implementation of the operations management discipline.

1.7.3 User Interface for Operations Staff

The user interface for operations staff visually represents managed objects and tracks agents on the network. Iconic images represent the managed devices, location, and status of the network environment, and these visual representations create a mirror world and permit a virtual roaming through the network. Query dialogs are provided to view information in the management database.

Figure 1-11 Operations Management Components

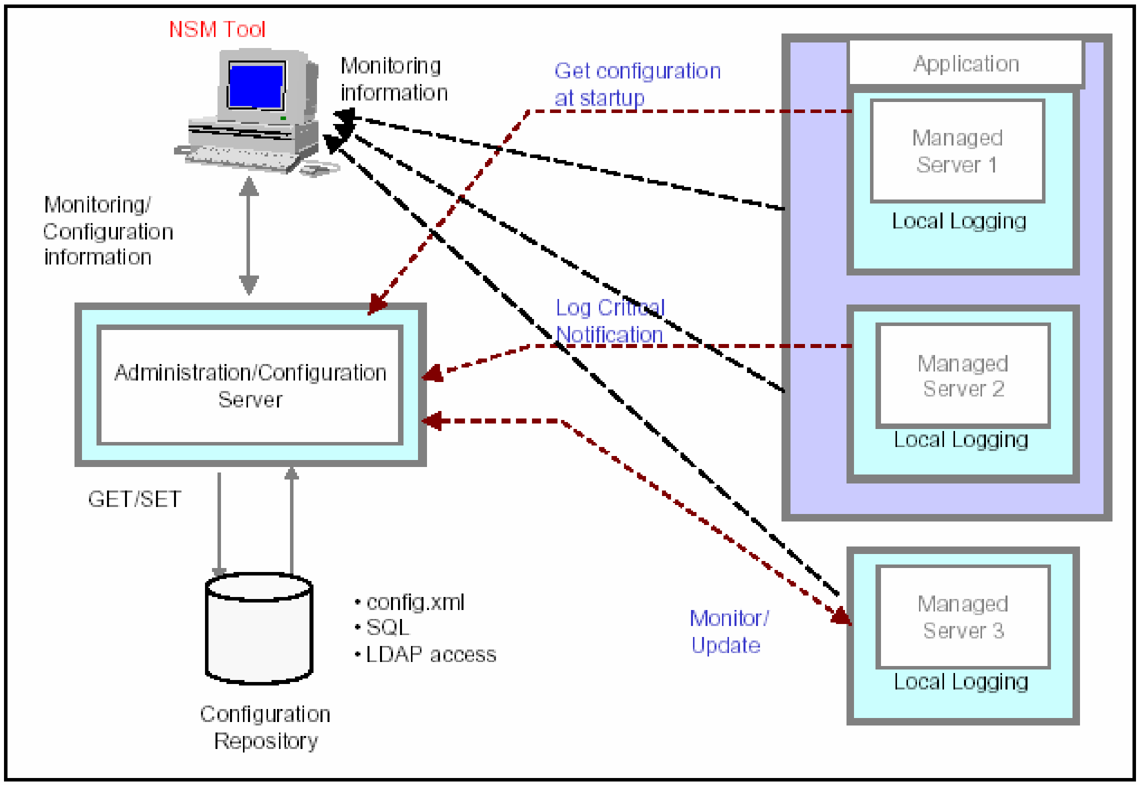


1.7.4 System Management Applications

Management applications perform systems management functions including operations, scheduling, configuration, problem and change and asset management. This is illustrated in Figure 1-11, Operations Management Components. They collect real-time information from the system components. These applications continuously monitor the system for potential problems and automatically launch corrective or preventive actions. They communicate with agents and other systems management components through high-level application interfaces (APIs).

A network systems management (NSM) framework is vendor-provided middleware that integrates multiple management applications through the use of collective services accessed through APIs. In an attempt to integrate management functions, vendors are providing product suites. No single product suite addresses all management functions; however, use of a suite provides a level of product integration.

Figure 1-12 Application Server Monitoring Scenario



(Adapted from: Application Server Management: Requirements, COM-17-1664, 6 August 2002.)

1.7.5 Management Information Database

The management information database is composed of information collected from the agents under the control of managing workstations. It is currently implemented using a relational database management system, RDBMS. Agents reside on the different managed network entities and continuously report on their status. Managing workstations can locate software agents anywhere on the network to gather management data and trigger responses to events. Real-time management data is maintained by the agents and stored on the local nodes they manage called management information bases (MIBs), as illustrated above in Figure 1-12, Application Monitoring Scenario.

1.7.6 Operations Architecture Technical Principles

The Operations Architecture domain is primarily directed by IT Technical Guiding Principles #6 and #10.

6. IT will focus on and be measured by the value of solutions delivered to internal and external stakeholders, and Transit customers.

10. Apply open systems concepts to insure portability, scalability, interoperability, and compatibility of information technology systems.

Because of the many complex areas within the Operations Architecture domain, additional domain-specific principles, technical guidelines, and standards are provided below as candidates for MDT IT Division’s adoption to assist in operations management.

Operations Principle OP.1: MDT IT will centralize remote systems management for mission critical applications.
Rationale:

- Systems management components are infrastructure and can be leveraged to provide common functionality to multiple business functions.

Operations Principle OP.2: Implement products that use standard protocols and interfaces.
Rationale

- Use of standard protocols and interfaces promotes interoperability among management products and can reduce required core management infrastructure, e.g., management consoles and products.

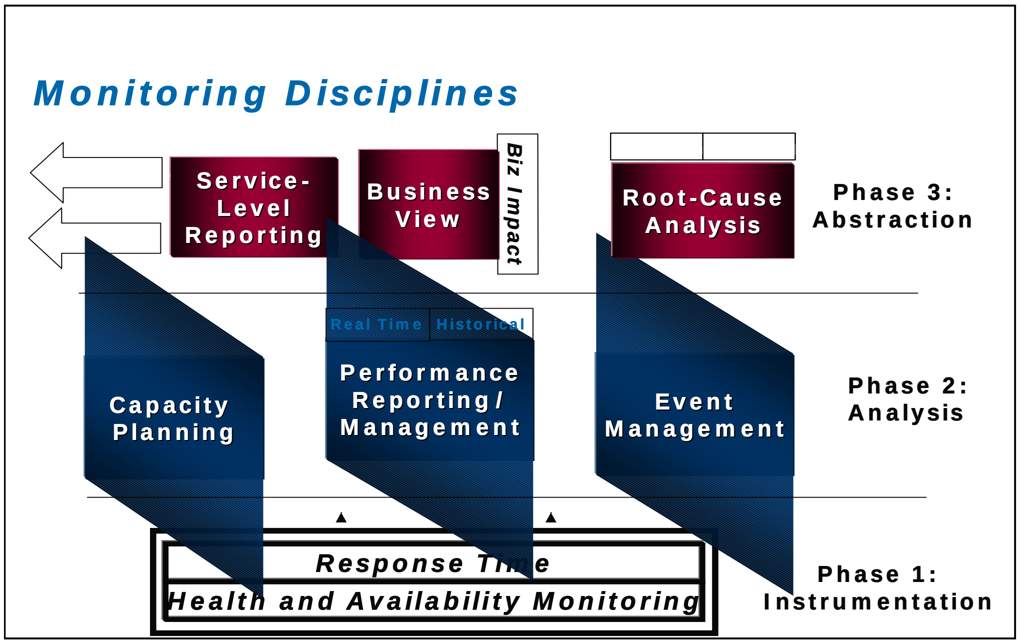
Operations Principle OP.3: MDT IT will deploy and leverage integrated management suites and only deploy best-of-breed point products when the integrated management suite cannot substantially meet business requirements.
Rationale:

- Investment and deployment of integrated tools reduces the costs associated with implementation and support. Common methods and procedures are used for bringing managed objects into the management framework and for ongoing management of these objects.

1.7.7 Operations Architecture Technical Guidelines

The three (3) phases of systems management capabilities and monitoring disciplines within the Operations Architecture environment are illustrated in Figure 1-13.

Figure 1-13 Operations Architecture Systems Management



(Adapted from: Application Server Management: Requirements, COM-17-1664, 6 August 2002.)

Additional guidelines, listed as OG.1, OG.2, etc., will better enable successful operations management:

- **Operations Guideline OG.1**: Equipment deployed within the data center or outlying offices will be configured to facilitate remote management and support.
- **Operations Guideline OG.2**: Equipment within the data center should be configured to prevent a single point of failure.
- **Operations Guidelines OG.3**: Common configurations of rack-mounted servers are placed in secure locations.
- **Operations Guidelines OG.4:** For reliability and ease of support, each major application should be placed on a uniformly configured server. This may require that each major application be implemented on its own server.
- **Operations Guidelines OG.5:** Deploy the same reference configuration on these servers. Considerations when planning for consistency include using the same versions of network software, using the same network hardware cards, etc.
- **Operations Guidelines OG.6:** Systems management tools, consistently applied, allow management of multiple instances of identical network configurations at remote sites as if they were on the data center floor.
- **Operations Guidelines OG.7:** Systems management functions for the data center should be remotely performed. Some examples of remote systems management services include:
    - Backup, archiving, and recovery

    - System, database, and application monitoring
    - Software distribution to the server and/or desktop
- **Operations Guidelines OG.8:** System components should proactively alert in advance of failure, including predictive capability. System-generated alarms and alerts should be automatically routed to the appropriate systems management resource.
- **Operations Guidelines OG.9:** Inventories of hardware and software configurations are critical to support functions and should be maintained in real time.
- **Operations Guidelines OG.10:** Inventory capability requires "agents" on workstations and servers.

1.7.8 Operations Architecture Standards

In addition to these guidelines the following standards support enterprise operational systems management. These are listed as OS.1, OS.2, etc.

Operations Standard OS.1: Use SNMP management protocols.
Rationale

- The Simple Network Management Protocol (SNMP) is a group of Internet protocols established as the standard for managing TCP/IP based networks.
- SNMP is integrated into the devices (e.g., concentrators, routers) of the network and in the network operating systems of the servers and workstations.
- Enterprise management systems use SNMP to collect statistics and other information on network devices.
- SNMP is also used to send commands that control the state of network devices.

Operations Standard OS.2: Use Remote Monitoring (RMON) products.
Rationale

- RMON products provide packet collection, decoding, and analysis stack using a combination of consoles and hardware and software probes that relied on SNMP MIB data collections.
- The RMON MIB extends SNMP capability by monitoring sub-network operation and reducing the data collection burden on management consoles and network agents.
- All major network device vendors have added RMON MIB collection capability to their products, although the depth of implementation relative to the full RMON specification varies among vendors and products.

Operations Standard OS.3: Conform to the Desktop Management Interface (DMI) standard.
Rationale

- The DMI standard was developed by the Desk Top Management Task Force (DMTF), which sets specifications for the management of the desktop environment.
- The DMI is a set of APIs that allows different vendor applications to consistently share the desktop.
- DMI sets the standard for a management platform that enables a common standardized mechanism for systems management of the desktop while permitting vendor differentiation.

2 WIRELESS AND ON-BOARD COMMUNICATIONS

The on-board physical/technical architecture may be viewed as a Reference Architecture for Metrobus, Metrorail, and Metromover. Details related to the architecture implementation will vary depending on the mode and fleet type.

These designs have been in use for some time in Transit vehicles in the fly-by-wire drive trains that move them. The VAN is a separate implementation that integrates all Information-level functions that are unique to Transit Operations. The VAN provides for the capture and real-time forwarding of drive train performance data, and some form of security software as well.

2.1 Vehicle Area Network

2.1.1 Description

"A data communications network that is installed on a public transit vehicle. The network supports data exchanges between various on-board subsystems. Similar to a Local Area Network (LAN) in a company or an office, a vehicle area network supports data communications primarily in a transit vehicle." [NTCIP 1406, p., 2-1]

2.1.2 Requirements

A communications network that controls and integrates the "Information" and the Input/Output (I/O) devices and subsystems installed on board the transit vehicle operating in revenue service. This "Information" must be clearly defined to fixed-end systems.

2.1.3 Available Standards

A Society of Automotive Engineers (SAE) family of standards defines the Vehicle Area Network (VAN), including:

- J-1708 – Protocol, error detection and correction
- J-1587 – Data and Metadata
- J-1455 – Environmental conditions in which vehicle devices must survive
- J-2496 – Cabling of a vehicle to permit plug-and-play insertion/removal of devices and fault diagnostics
- J-1939 – High speed communications (currently in final ballot at the SAE)

These designs have been in use for some time in Transit vehicles in the fly-by-wire drive trains that move them. The VAN is a separate implementation that integrates all Information-level functions that are unique to Transit Operations, as well as providing for the capture and real-time forwarding of drive train performance data.

2.2 Wireless Communications

2.2.1 Description

Wireless Communications is a means of transferring data to and from the vehicle while stationary or moving at low speed. Traditional radio systems (e.g., 800/900 MHz Trunked Voice/Data radio networks) are used for the transfer of real-time vehicle data (location, alarms, status, etc.) and management direction (driver instructions). This type of radio technology is too slow, and shared by too many vehicles, for the massive amounts of data required by Annunciator, AVL, and Schedule Adherence Subsystems on the vehicle. Likewise, a large amount of performance data collected on the vehicle needs to be transferred as batch data at the end of the day.

Wireless LAN technology and/or Digital Short-Range Communications (DSRC) standards are becoming the technology of choice for these large data transfers at specific points (i.e., transit garages, transit centers, park & ride lots, etc.).

2.2.2 Requirements

The transfer rate of wireless networks is normally far too high to be handled by serial communications ports (i.e., RS-232). The vehicle's Logic Unit (controller) would normally interface to these systems with Databus speed via Direct Memory Access (DMA), either directly or through a Network Interface Card (NIC). Older Logic Units may or may not be sufficient to accommodate the new wireless technology.

2.2.3 Available Standards

Currently, there are several standards that are worthy of consideration:

- IEEE 802.11a
- IEEE 802.11b
- IEEE 802.11g
- DSRC

It should be noted that these are transport-level standards. The data and metadata transported by these transport standards should consider the ITS TCIP standards for data content and message sets. Each of these standards requires some form of security software as well.

2.2.4 Wireless Communications Issues

A single wireless communication technology is desirable. This technology would be used for multiple on-board subsystems including video surveillance, loading configuration data, off-loading operational performance, and fare box/APC stop level ridership data. Security of the information remains uncertain. Since the wireless Ethernet technology (IEEE 802.11x) is a standard, it is open to attack by hackers. There are software or technology fixes to ensure authentication and access to on-board information that will block "enterprising college students" or others.

2.3 Radio Communications

2.3.1 Description

There has always been a need for real-time Dispatcher-to-Driver-to-Dispatcher communications in Transit systems. Originally, this communication was limited to voice only. Later, digital data enhanced radio networks enabled the exchange of AVL, alarm, and status information as well as streamlining instructions to drivers through alphanumeric control heads on vehicles. To date, there is no cost-effective alternative to these networks when large numbers of vehicles are involved.

2.3.2 Requirements

Real-time radio networks must provide for:

- At least 95% coverage of the service area 95% of the time
- Sufficient numbers of dispatcher positions to handle the driver-waiting queues
- A design capable of logically addressing the number of vehicles in the fleet
- Silent alarms from vehicles, even if all other on-vehicle subsystems fail
- Transport of real-time data to and from the vehicle and its on-board devices

2.3.3 Available Standards

There are no published standards associated with voice / data networks. Rather, each radio manufacturer has protected their years of research and development in proprietary designs with patents and copyrights (i.e., Motorola, Johnson, Ericsson, etc.). However, these networks can be viewed as a data transportation network that is capable of transparently passing data defined by published standards (i.e., SAE J-1587, TCIP) if the system integrator is charged with this requirement during procurement.